Программирование разветвляющихся алгоритмов

А.ШАКИРИН,
г.Минск, БАТУ, каф. ВТ.

Цель этой статьи — освоить использование простейших компонентов-переключателей и создать приложение, которое использует разветвляющийся алгоритм.

1. Пример создания приложения

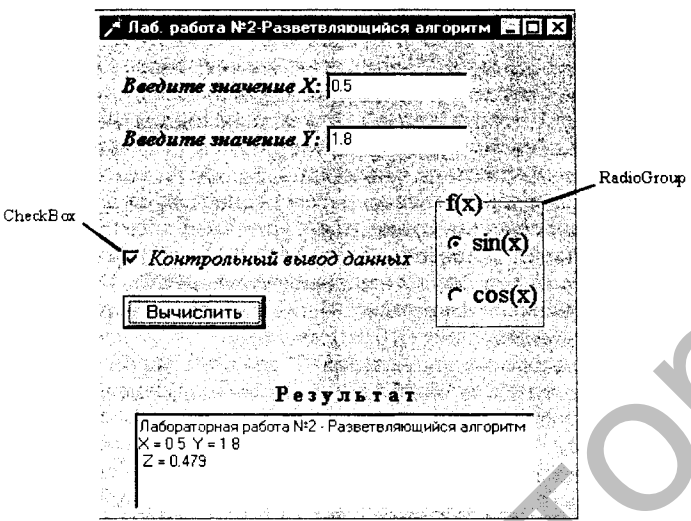
Необходимо создать Windows-приложение для вычисления выражения

$$Z = \begin{cases} f(x), x < y \\ y, \text{иначе} \end{cases},$$

где *f(x)=sin(x)*, либо *f(x)=cos(x)*. В панели интерфейса следует предусмотреть возможность управления контрольным выводом исходных данных.

1.1. Размещение компонентов на Форме

Будем размещать компоненты на Форме так, чтобы они соответствовали панели, показанной на рисунке.



При создании приложений в Delphi часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено-выключено) визуально отражается на Форме. На панели представлены кнопки-переключатели двух типов — CheckBox и RadioGroup.

Компонент **CheckBox** организует кнопку независимого переключателя, с помощью которой пользователь может указать свое решение типа "да/нет".

Компонент **RadioGroup** организует *группу кнопок* — зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки выключаются.

Поместите на Форму компоненты Label, Edit и Memo в соответствии с рисунком. Выберите в Палитре Компонентов на странице Standard пиктограмму компонента CheckBox и разместите ее в нужном месте Формы. В свойстве Caption Инспектора Объектов замените надпись **CheckBox1** на ***Контрольный вывод данных***. Чтобы при запуске приложения кнопка CheckBox оказалась включена, свойство Checked установите равным True.

Выберите в Палитре Компонентов Standard пиктограмму компонента RadioGroup и поместите ее в нужное место Формы. В свойстве Caption измените заголовок **RadioGroup1** на **f(x)**. Для размещения кнопок в один столбец, свойство Columns установите равным 1. Дважды щелкните мышью по правой части свойства Items — появится строчный редактор списка наименований кнопок. Наберите две строки с именами: в первой строке — sin(x), во второй — cos(x), и нажмите OK. После этого на Форме появится группа из двух кнопок — переключателей с соответствующими надписями. Чтобы при запуске приложения первая кнопка RadioGroup оказалась включена, свойство ItemIndex установите равным 0.

1.2. Создание процедур обработки событий FormCreate и Button1Click

Технология создания процедур обработки событий FormCreate и Button1Click ничем не отличается от описанной в [1]. Внимательно наберите операторы этих процедур, используя текст модуля UnRazvAlg.

1.3. Текст модуля UnRazvAlg

```
Unit UnRazvAlg;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label4: TLabel;
    Memo1: TMemo;
    Button1: TButton;
    RadioGroup1: TRadioGroup;
    CheckBox1: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}
// Процедура обработки события создания Формы:
procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:='0.5';      // начальное значение X
 Edit2.Text:='1.8';      // начальное значение Y
 Memo1.Clear;            // очистка  Memo1
// Вывод строки в  Memo1:
 Memo1.Lines.Add('Лабораторная работа №2 - Разветвляющийся '+
'алгоритм');
end;
// Процедура обработки события нажатия кнопки Button1:
procedure TForm1.Button1Click(Sender: TObject);
 var
  x,y,z,fx : extended;      // объявление локальных переменных
begin
 x:=StrToFloat(Edit1.Text); // X присваивается содержимое Edit1
 y:=StrToFloat(Edit2.Text); // Y присваивается содержимое Edit2
 fx:=sin(x);                // fx присваивается начальное значение
// Выбор функции, соответствующей нажатой кнопке:
 case RadioGroup1.ItemIndex of
  0:fx:=sin(x);
  1:fx:=cos(x);
 end;
 // Вычисление выражения:
 if x<y then
         z:=fx
         else
         z:=y;
// Проверка состояния кнопки CheckBox1:
  if CheckBox1.Checked then
                    Memo1.Lines.Add('X = '+Edit1.Text+
```

Репозиторий БГАТУ

```
  '  Y = '+Edit2.Text);    // контрольный вывод X, Y, Z в Memo1
                           // Вывод результата в Memo1:
 Memo1.Lines.Add(' Z = '+FloatToStrF(z,ffFixed,8,3));
end;
end.
```

Если нажата первая кнопка RadioGroup1, в переменную целого типа RadioGroup1.ItemIndex заносится ноль, если вторая — единица. Если кнопка CheckBox1 нажата, логическая переменная CheckBox1.Checked имеет значение True, если нет — False.

1.4. Работа с приложением

Запустите созданное приложение. Используя все управляющие компоненты панели интерфейса, убедитесь в правильном функционировании приложения во всех предусмотренных режимах работы.

2. Индивидуальные задания

Необходимо создать Windows-приложение для вычисления соответствующего выражения и протестировать его работу. На панели интерфейса следует предусмотреть возможность выбора одной из трех функций f(x): sh(x), x^2, e^x.

1. $a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & x\le 0\\ (f(x)+y)^3+\sqrt{|f(x)y|}, & 0<x\le 1\\ (f(x)+y)^4+1, & \text{иначе.}\end{cases}$

2. $b=\begin{cases}ln(f(x))+\left(f(x)^2+y\right)^3, & x/y>0\\ ln|f(x)/y|+(f(x)+y)^2, & x/y<0\\ \left(f(x)^3+y\right)^2, & -2\le x<2\\ 0, & \text{иначе.}\end{cases}$

3. $c=\begin{cases}f(x)^2+y^2+sin(y), & x+y>0\\ \left(f(x)-y^3\right)^2+cos(y), & x>0,\ y<0\\ (y-f(x))^2+tg(y), & x-y<0.\end{cases}$

4. $d=\begin{cases}f^2(x)+arctg(f(x)), & 1\le x<5\\ (y-f(x))^2+arctg(f(x)), & y>x\\ (y+f(x))^3+0.5, & \text{иначе.}\end{cases}$

5. $e=\begin{cases}\sqrt[i]{f(x)}, & i-\text{нечетное}, x>0\\ i/2\sqrt{|f(x)|}, & i-\text{четное}, \quad x<0\\ \sqrt{|if(x)|}, & \text{иначе.}\end{cases}$

6. $g=\begin{cases}e^{f(x)-|b|}, & 0.5<xb<10\\ \sqrt{|f(x)+b|}, & 0.1<xb<0.5\\ 2f(x)^2, & \text{иначе.}\end{cases}$

7. $s=\begin{cases}e^{f(x)}, & 1<xb<10\\ \sqrt{|f(x)+4\cdot b|}, & 12<xb<40\\ bf(x)^2, & \text{иначе.}\end{cases}$

8. $j=\begin{cases}sin\left(5f(x)+3m|f(x)|\right), & -1<m<x\\ cos\left(3f(x)+5m|f(x)|\right), & x\le m\\ (f(x)+m)^2, & \text{иначе.}\end{cases}$

Литература

1. А.Шакирин. Программирование линейных алгоритмов. — Радиомир. Ваш компьютер, 2001, №8, С.21.

2. В.В.Феофанов. Delphi 3. Учебный курс. — М.: Нолидж, 1981.

3. Э.Возневич. Delphi. Освой самостоятельно. — М.: Восточная книжная компания, 1996.

4. Дж.Матчо, Д.Р.Фолкнер. Delphi. — М.: БИНОМ, 1995.

5. М.Канту. Delphi 2 для Windows 95/NT. — М.: ООО "Малип", 1997.

HI-TECH,
E-mail: serrgio@gorki.unibel.by

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

Обращение к читателям

— Что может сказать собака о сущности дзена?
— Гав, — скажет собака.
Одна из многочисленных дзенских баек

Киньте грязью в того, кто вам скажет, что Ассемблер — очень сложный для изучения язык. И никогда не читайте глупых книг, в которых написана подобная чушь. О том, что это очень сложно, говорят и пишут люди, у которых в свое время не хватило смелости (и/или способностей) попытаться "въехать" в "машинные коды", "прерывания", "порты ввода-вывода" и прочую низкоуровневую "чепуху", с которой рано или поздно сталкивается любой профессиональный программист. Можно сколько угодно ругать глюки в "винде", кривой SQL в Delphi, Билла Гейтса или "эту проклятую зидовскую мамку" — это не избавляет от элементарного невежества в области "компьютерных технологий".

Мы, авторы этого курса — "Низкоуровневое программирование для дZенствующих" — считаем, что принципы функционирования компьютера и некоторые основы низкоуровневого программирования (наверное, последнее точнее будет назвать "кодированием") относятся к категории элементарных знаний, владеть которыми обязан КАЖДЫЙ программист.

В общем, если у вас есть желание изучить ЭТО — читайте дальше. Был бы ученик толковый, а уж учителями мы постараемся стать хорошими :).

Короче, добро пожаловать в мир, где программист — хозяин компьютера, а не наоборот; в мир низкоуровневого программирования.

Что такое "низкоуровневое программирование", наверное, понятно, но почему оно "для дZенствующих"?

Такое название выбрано по нескольким причинам.

1. "По приколу" (с нашим плоским компьютерным юмором вы будете сталкиваться постоянно).

2. Потому что программирование для нас — намного больше, чем просто программирование. Во-первых, низкоуровневое программирование — это состояние души. Во-вторых, это состояние сознания; использование некоторых медитативных приемов для настройки на рабочий лад; иногда — состояние транса (например, при отладке программы); и в конце концов — полная нирвана (когда, наконец, написанная на чистом asm'е программа — работает!).

3. А теперь самая главная причина. Авторы проекта — люди дZенствующие, "по жизни". Понимайте это как хотите...

Материал, который мы вам предлагаем — это адаптированная и слегка переработанная версия одноименной рассылки, которую мы периодически распространяем через сервисы subscribe.ru и maillist.ru (7500 подписчиков).

Весь этот "бред" (самокритика) был протестирован на "полных чайниках" и доведен до такой степени "удобоваримости", что, чтобы не понять его, нужно уж очень постараться.

Скажем сразу, определенный круг лиц был недоволен нашим способом и стилем "преподношения" материала. Они говорили, что это некрасиво, что это несерьезно, что это непедагогично... Это в основном были люди старшего поколения, "творящие" свои произведения по принципу — "нужна публикация". Это были люди, часто имеющие смутное представление о программировании, но весьма успешно подтверждавшие свою "компетентность" обилием ссылок на различные "авторитетные" источники, преемствующие, наряду с опытом предшественников, также и их технические ошибки... :(

Репозиторий ГГТУ